Js Sample Exam Answer

Question 1: Answer the following questions :

1. What is an Event Loop?

Ans: An event loop is a core programming construct that facilitates asynchronous programming. It allows a program to handle multiple operations simultaneously without blocking the main execution thread. This is particularly important in environments where responsiveness and efficient handling of events are critical, such as in web browsers and servers.

1. How do you add an element at the begining of an array? How doyou add one at the end? Hint[ 2 ways ]

Ans: To add an element at the beginning of an array: Use unshift(element).

To add an element at the end of an array: Use push(element).

Question 2: What is The Output ?

1. Output: 3
2. Output:
3. // after 0 seconds
4. // after 1 second

2 // after 2 seconds

3 // after 3 seconds

4 // after 4 seconds

1. Output: ["baz"]
2. Output:

1

Hello

true

1. Output:

true

[ [ 'carName', 'Bmw' ], [ 'carPrice', 1000000 ] ]

Question 3:

1- Create Function sumObjectValues() that will sum all values of

the fields that contain numbers. Ensure that iteration is

done only over own property of the object.

Ans:

function sumObjectValues(*obj*) {

    let sum = 0;

    Object.keys(obj).forEach(*key* => {

        if (typeof obj[key] === 'number') {

            sum += obj[key];

        }

    });

    return sum;

}

*// Example usage*

const object = {

    a: 10,

    b: 20,

    c: "string",

    d: 30

};

console.log(sumObjectValues(object)); *// Output: 60*

2- Show the execution of 3 asynchronous block of code, one after

the other in sequence.

Ans:

*// Simulate an asynchronous operation with a promise that resolves after a delay*

function asyncOperation(*message*, *delay*) {

    return new Promise(*resolve* => {

        setTimeout(() => {

            console.log(message);

            resolve();

        }, delay);

    });

}

*// Define an async function to run the operations in sequence*

async function executeInSequence() {

    await asyncOperation('First operation completed', 1000);  *// 1 second delay*

    await asyncOperation('Second operation completed', 2000); *// 2 seconds delay*

    await asyncOperation('Third operation completed', 1500);  *// 1.5 seconds delay*

}

*// Run the sequence*

executeInSequence();

1. Get the maximum value from a numbers array along with its index.

Ans:

function getMaxValueAndIndex(*numbers*) {

    if (numbers.length === 0) {

        return null; *// Handle empty array case*

    }

    let max = numbers[0];

    let maxIndex = 0;

    for (let i = 1; i < numbers.length; i++) {

        if (numbers[i] > max) {

            max = numbers[i];

            maxIndex = i;

        }

    }

    return { max, maxIndex };

}

*// Example usage*

const numbersArray = [5, 2, 9, 1, 5, 6];

const result = getMaxValueAndIndex(numbersArray);

console.log(`Maximum value: ${result.max} at index ${result.maxIndex}`);

1. Write a function which accepts two valid dates and returns the

difference between them as number of days.

Ans:

    function differenceInDays(*date1*, *date2*) {

*// Calculate the difference in milliseconds*

        const diffInMs = Math.abs(date2.getTime() - date1.getTime());

*// Convert milliseconds to days*

        const diffInDays = Math.floor(diffInMs / (1000 \* 60 \* 60 \* 24));

        return diffInDays;

    }

*// Example usage*

    const startDate = new Date('2024-01-01');

    const endDate = new Date('2024-07-01');

    const daysDifference = differenceInDays(startDate, endDate);

    console.log(`Difference in days: ${daysDifference}`);

// Output : Difference in days: 182

1. Design a Calulator interface for 2 number inputs which can

perform sum, difference, product and dividend whenever invoked

on the same interface.

Ans:

 function performOperation(*operation*) {

    function add(*num1*, *num2*) {

  return num1 + num2;

}

function subtract(*num1*, *num2*) {

  return  num1 - num2;

}

function multiply(*num1*, *num2*) {

  return num1 \* num2;

}

function divide(*num1*, *num2*) {

*// Handle division by zero*

  if (num2 === 0) {

    return "Error: Division by zero";

  }

  return num1 / num2;

}

return {add,subtract,multiply,divide}

 }

const calc = performOperation(); *// Call performOperation to get the object*

console.log(calc.add(3, 4));

console.log(calc.subtract(3, 4));

console.log(calc.multiply(3, 4));

console.log(calc.divide(3, 4));

1. Write a function which can return multiple values from a function.

Ans:

function calculateAreaAndPerimeter(*width*, *height*) {

  const area = width \* height;

  const perimeter = 2 \* (width + height);

  return {area, perimeter}; *// Return an array containing both values*

}

1. Write a function to reverse an array. For Example:

reverse ([1, 2, 3, 4]) ➞ [4, 3, 2, 1]

Ans:

function reverseArr(*input*) {

    var arr = [];

    for(var i = input.length-1; i >= 0; i--) {

        arr.push(input[i]);

    }

    return arr;

}

var a = [3,5,7,8]

var b = reverseArr(a);

console.log(b)

1. Write a function that converts an object into an array, where each

element represents a key-value pair in the form of an array.

For Example:

({a: 1, b: 2}) ➞ [["a", 1], ["b", 2]]

Ans:

function objectToArray(*obj*) {

*// Use Object.entries to get an array of key-value pairs*

  const entries = Object.entries(obj);

  return entries;

}

*// Example usage*

const myObject = { a: 1, b: 2 };

const keyValueArray = objectToArray(myObject);

console.log(keyValueArray);  *// Output: [["a", 1], ["b", 2]]*

Bonus ++

1- Write a function which can convert the time input given in 12

hours format to 24 hours format.

Ans:

function convertTo24HourFormat(*time12h*) {

    let [time, modifier] = time12h.split(' ');

    let [hours, minutes] = time.split(':');

    if (hours === '12') {

        hours = '00';

    }

    if (modifier === 'PM') {

        hours = parseInt(hours, 10) + 12;

    }

    return `${hours}:${minutes}`;

}

*// Example usage:*

console.log(convertTo24HourFormat("02:30 PM")); *// Output: "14:30"*

console.log(convertTo24HourFormat("12:45 AM")); *// Output: "00:45"*

2- Make this syntax possible: var a = add(2)(3); //5

Ans:

function add(*x*) {

    return function(*y*) {

        return x + y;

    };

}

*// Example usage:*

var a = add(2)(3);  *// 5*

console.log(a);     *// Output: 5*

3- Check if the user with the name "John" exists in the array of

objects.

Ans:

*// Example array of user objects*

const users = [

    { name: "Alice", age: 25 },

    { name: "Bob", age: 30 },

    { name: "John", age: 35 },

    { name: "Jane", age: 28 }

];

*// Check if a user with the name "John" exists*

const userExists = users.some(*user* => user.name === "John");

console.log(userExists); *// Output: true*